Embedded Applications Journal

A PUBLICATION OF INTEL CORPORATION SECOND QUARTER, 1994

What's Inside

Feature Articles

Errata and Change Identifiers

Introducing the 8XC196NP

Rick Evans
Applications Engineer
Intel Corporation
Article ID# 0801

This article provides an overview of the 8XC196NP. For a full list of features and specifications, please order the 8XC196NP datasheet (order number 272459) and quick reference guide (order number 272466). For additional information, request the user's manual (order number 272479).

We are proud to announce the newest member of the MCS® 96 microcontroller family, the 8XC196NP. This device operates at 25 MHz at 5 volts and 16 MHz at 3.3 volts. The 8XC196NP has 20 address lines for addressing up to 1 Mbyte of address space, and is available with 4 Kbytes of ROM or no ROM. This new device can be used for those applications that are currently at 5 volts and going to low voltage or as a performance upgrade to 25 MHz at 5 volts.

In addition to its performance and low power features, the 8XC196NP has a demultiplexed address/data bus, which allows the use of slower memory devices. In fact, the 8XC196NP can run at 25 MHz with zero wait states with a 100 ns memory device. The address/data bus can be dynamically switched between multiplexed and demultiplexed operation. It also has a chip-select unit, which simplifies memory interfacing. The chip-select unit has six separate outputs, each of which can select an address range of 256 bytes up to 1 Mbyte. Moreover, each chip-select can be configured for 8- or 16-bit buswidth and for 0 to 3 wait states. That eliminates the need for designing a wait-state generator.

Let's look at an example memory system to see the advantages of the demultiplexed bus and the chip-select unit. We'll hook up three devices: a Flash, an SRAM, and an external UART. This requires three chip-selects. (See Figure 1.)

Each chip-select has three registers: an address mask (ADDRMSKx), an address

Continued on page 3

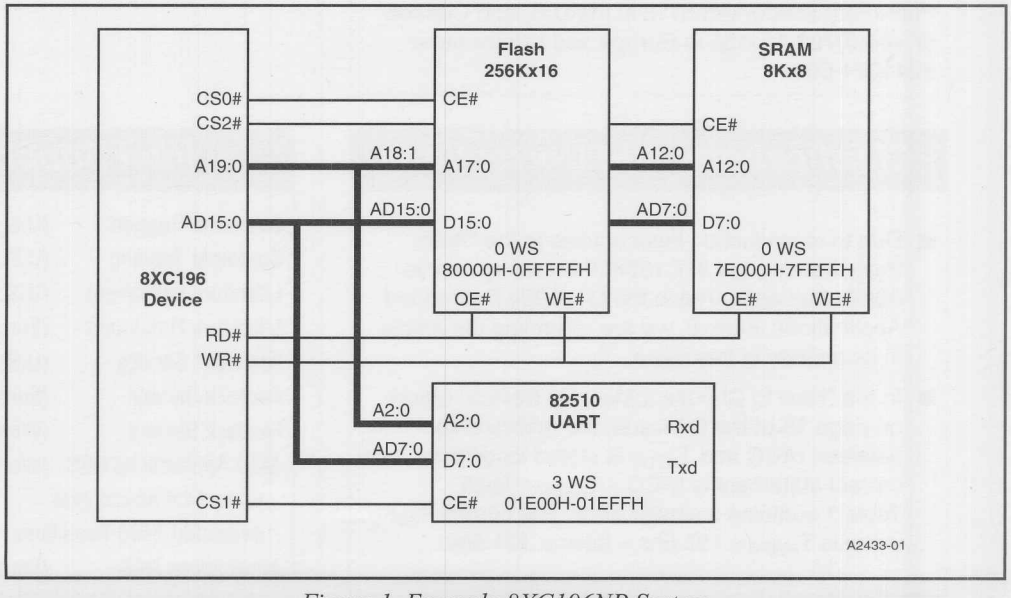

Figure 1. Example 8XC196NP System

From the Managing Editor

Over this past year, we have received overwhelming, positive response to the articles in the Embedded Applications Journal (EAJ). We are happy to provide such quality. Since 1992 we have been featuring articles on programming, 3V/5V mixed voltages, introductions of new embedded devices, fuzzy logic, and overall functionality of our microcontrollers and microprocessors.

Starting next quarter, we would like to feature your design ideas for Intel's embedded microcontrollers and microprocessors, so we're running a contest. Fax us your design ideas. We need to know:

1. your name, address, and phone number
2. your design idea (block diagrams, schematics, etc.)
3. what Intel devices you are using (don't forget Flash memory and FLEXlogic)
4. what is unique about your design (no proprietary information)
5. how the design works (description of the basic functionality)

This effort will not go unrewarded. In addition to having your idea published in the following quarter's EAJ, the single best idea will be rewarded with an Intel SatisFAXtion 14.4Kbps modem/400e external fax/modem plus Intel's FAXability Plus/OCR software — the perfect item for keeping in touch with Intel's BBS and FaxBack service.

So start faxing in your ideas today!

1-800-722-2862 in the U.S. and Canada or 1-602-554-7436 worldwide.

Don't forget — April 1994 starts the world tour of the Embedded Solutions Seminars (ESS II) in 29 cities worldwide. To register, see your local Intel distributor or call 1-800-368-4110 in North America or +44(0) 793-431155 for the European Seminars.

Steven M. McIntyre
Embedded Applications Manager
Embedded Microcomputer Division

More Copies

Need more copies of this issue of the Embedded Applications Journal? Please call Intel Literature Fulfillment at 800-468-8118 in the U.S. and Canada or +44(0)793-431155 in Europe and ask for order #241294-008.

Clarifications and Corrections

- Due to reproduction inaccuracies in the "Auto Programming the 87C196KD" article by Jennie Abella that appeared in the Q1, 1994 Embedded Applications Journal, we are rerunning the article in its entirety in this issue.
- In the "How to Choose a Memory Device" article on page 18 of the Q1 issue, the relationship between tACC and T_{AVDV} is stated incorrectly. The correct statement is tACC ≤ T_{AVDV} – latch. Table 1 contains a similar error. The correct T_{AVDV} value is T_{AVDV} = 132.5ns – latch = 121.5ns.

Intel Support Numbers

Customer Support	(U.S. and Canada)	800-628-8686
Customer Training	(U.S. and Canada)	800-234-8806
Literature Fulfillment	(U.S. and Canada)	800-468-8118
Literature Fulfillment	(Europe)	+44(0)793-431155
FaxBack* Service	(U.S. and Canada)	800-628-2283
FaxBack Service	(Europe)	+44(0)793-496646
FaxBack Service	(Worldwide)	916-356-3105
AMO Applications BBS	(Worldwide)	
up to 14.4 Kbaud lines		916-356-3600
dedicated 2400-baud lines		916-356-7209
Applications BBS	(Europe)	+44(0)793-496340

Bit Number(s)	Bit Mnemonic	Description
7	MS	Mode Select: This bit selects the external bus mode. 1 = Demultiplexed 0 = Multiplexed
6	BW	Buswidth: This bit selects the external bus width. 1 = 16-bit bus 0 = 8-bit bus
5–2	—	Reserved; always write zero to these bits for future compatibility. Reading a reserved bit produces an indeterminate result.
1–0	WS1–WS0	Wait States: These bits control the number of wait states. **WS1 WS0 Number of wait states** 0 0 0 0 1 1 plus any requested by READY pin 1 0 2 plus any requested by READY pin 1 1 3 plus any requested by READY pin

Figure 2. Bus Control Register, BUSCONx (x=0-5)

Continued from page 1

compare (ADDRCOMx), and a bus control register (BUSCONx). The ADDRMSKx and ADDRCOMx select the address range. The BUSCONx register (Figure 2) configures the bus for each particular chip-select, so it is very easy to configure the bus for each address range.

For the system in Figure 1, we eliminated two latches and a logic device (EPLD or PAL). The logic device would have had to decode the chip-select, generate wait states, and provide feedback to a buswidth pin. The 8XC196NP has no need for a buswidth pin, since the BUSCONx register controls the buswidth.

Summary

The 8XC196NP is a good fit for low-power portable applications or those applications that require more performance at 5 volts. It is available in a 100-pin SQFP or QFP package.

The Use of the iRMX® EMB Software Simulator in Intel386™ Processor System Design

Tom Lehmann
Technical Marketing Manager
iRMX Operating System Products
Article ID #0802

Abstract

The use of the simulator included with the iRMX® EMB Software Development Kit will shorten the design/development cycle by providing an accurate test vehicle before final working hardware can be created.

Introduction

The development of a typical PC-based system can usually proceed without software development delay because final system test hardware is usually available almost from the beginning. In contrast, development of a system based on custom hardware (as would be the case with an Intel386™ EX processor-based design) can involve significant software development delays due to the unavailability of target hardware or an inaccuracy in the target hardware specification. The simu-

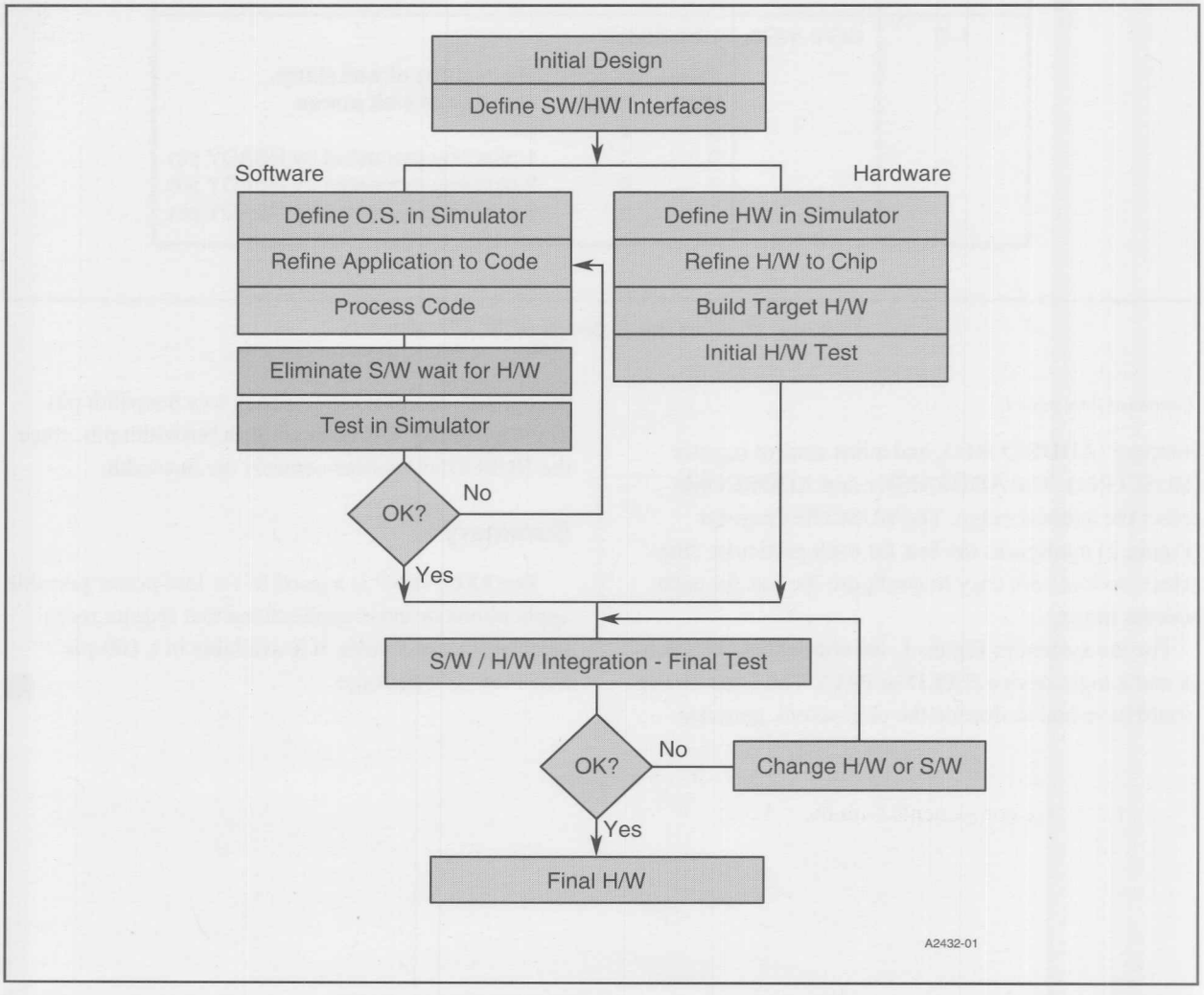

Figure 1. Development Process Using a Simulator

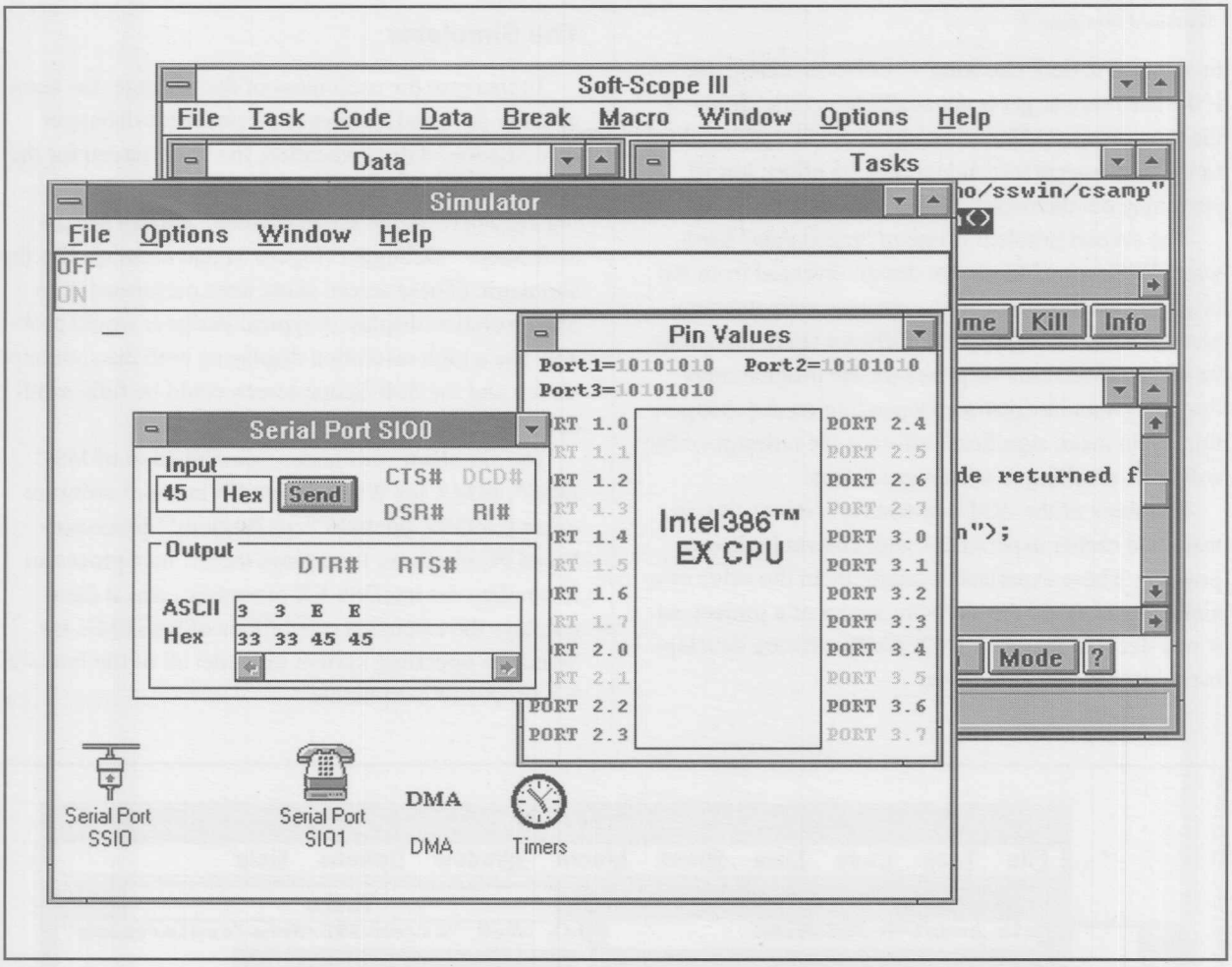

Figure 2. iRMX® EMB Software Simulator for Intel386™ EX Processor

lator included with the iRMX EMB Software Development Kit should help avoid or significantly reduce these problems. This article will describe the simulator and give a brief example of its usage within the development cycle.

The Design Process

The overall design cycle (top time line in Figure 1) is broken roughly into the following steps:

1. Initial, high level design of the system (the 50,000 foot view)
2. Break down the system functionality into hardware and software components with the definition of the hardware-software interfaces.
3. Team proceeds with the simultaneous refinement of design details followed by the simultaneous development of code and hardware.
4. The software development team can only "desk check" their code until some target hardware becomes available.
5. When test hardware becomes available, software testing can proceed. Depending on the size of the software, more than one test target may be necessary to allow software testing to proceed unhindered.
6. As inconsistencies (bugs) in hardware and software are discovered, changes are made in one or both until final testing is completed satisfactorily.
7. The final system is constructed and production can begin.

The Two Possible Problems

As mentioned earlier, there are two possible problems with the non-simulator approach in a custom hardware-based project. The software development team can

Continued on page 6

Continued from page 5

be reduced to desk checking — or worse, sitting idle — if the hardware target is not available in time. Some of the time they could have been testing will then be added to the final system test and integration effort, further stretching out the overall project development.

The second problem is one of "inaccurate" hardware. If the actual hardware design diverged from the original specification and the changes were not completely communicated to the software team, there may be some unwelcome surprises for the programmers during system integration. Depending on the changes, this could mean significant effort in the redesign of the software, resulting in even more delays.

Members of the iRMX operating system's design team had earlier experiences with custom hardware projects. These experiences taught them the value of a simulator to speed the software aspect of a project, so it was decided that the iRMX EMB software development kit should contain one.

The Simulator

To increase the usefulness of the package, the team actually designed an integrated simulator/debugger combination. After invocation, the main screen for the simulator can be seen in Figure 2. Note that by clicking anywhere on the second screen, you can see the Soft-Scope* Debugger (Figure 3) that accompanies the simulator. (These screen shots were performed on a low resolution display. A typical designer would probably use a high resolution display so both the simulator screen and the Soft-Scope screen could be fully available during the debugging session.)

The simulator runs under a combination of MS-DOS*, iRMX for Windows, and Windows* software on an Intel386, Intel486™, or Pentium™ processor-based PC platform. It employs the PC host processor to emulate the Intel386 EX processor core. It then employs the extension capabilities of the iRMX for Windows operating system to model all of the Intel386 EX processor peripherals.

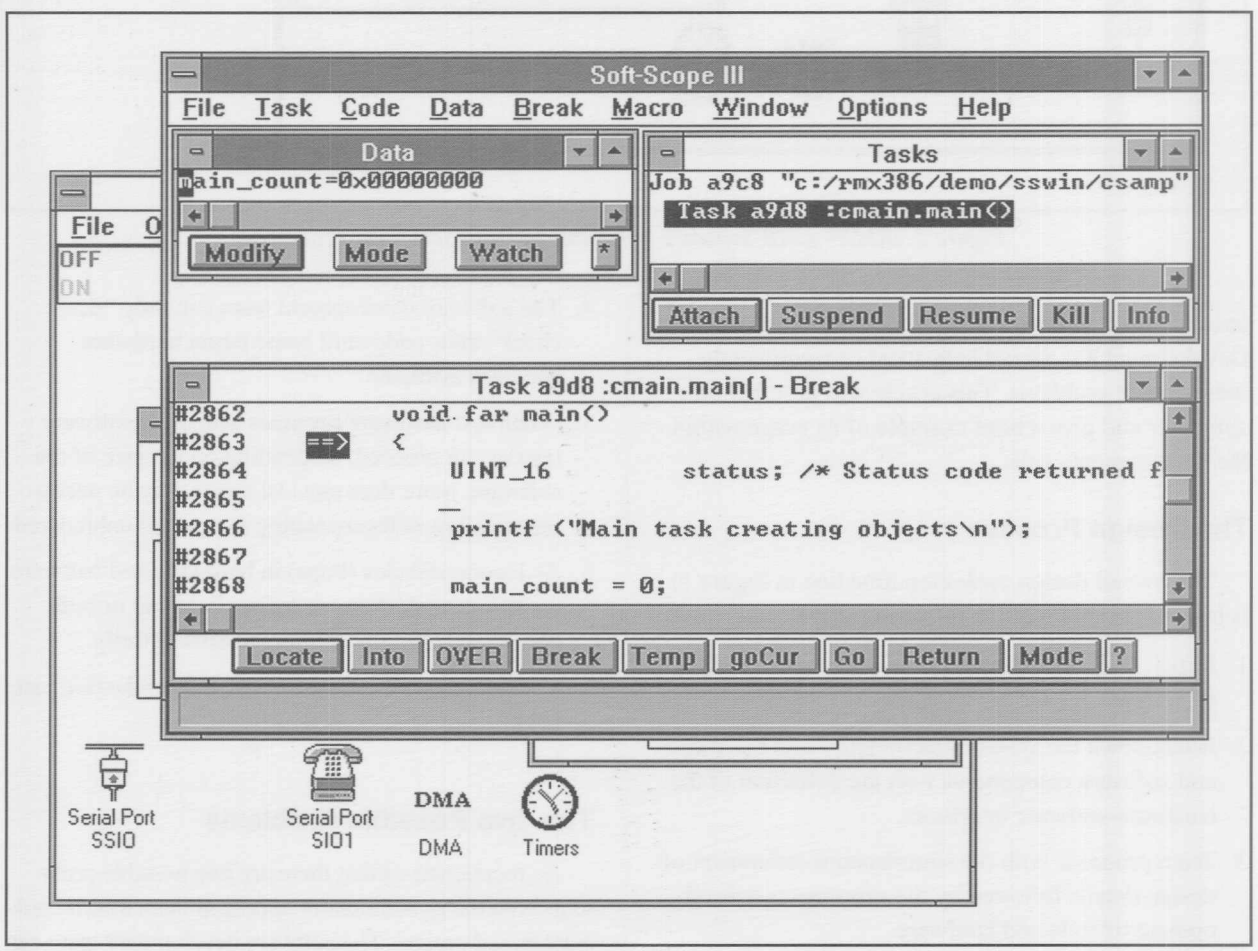

Figure 3. Soft-Scope Debugger*

Note: *This is a functional simulator. It does not run in real time. Core processor instructions may run much faster in the simulator (especially on a 66 MHz Pentium processor-based machine), while simulated peripherals could run much slower (modeled from disk files, for instance).*

To accomplish all this "magic," iRMX for Windows software, the simulator, and the Soft-Scope debugger are all running in ring 0 of the protection model. The simulated application, sometimes referred to as "guest code," is run in Ring 3. This allows the iRMX for Windows software and simulator combination to automatically trap all guest code I/O requests and direct them as desired. The pattern of the redirection is accomplished through a Windows software-based Intel Configuration Utility (ICU). This program is used to configure the operating system, the chip and its peripherals, and the simulator all at the same time.

Configuration Files

The ICU produces a configuration file, <userdefined name>.ICU. This file, known as the ICU file, is composed of many sections, but for now we will only be concerned with the following five sections: operating system, device configuration, device simulation, external world stimulation, and recording.

The operating system entries define information that the developer selects to describe the target operating system. The simulator uses this information to ensure that an application program does not use any facility of the simulator that will not be present in the final system. For example, in order to limit memory requirements of the final target system, the developer may decide to eliminate semaphore operations. Knowing this, the simulator would notify the developer of the illegal operation if an application program attempted to create a semaphore.

The device configuration entries describe the configuration of the peripherals for this system. This information is used to set up the simulator. It also configures the operating system to correctly program the peripherals during system operation.

The device simulation portion of the file describes the actions to be taken by the simulator upon intercepting any I/O requests. At the moment the four choices are to:

a) Ignore the request

b) Redirect the I/O to PC host I/O (this is especially easy with serial ports)

c) Simulate the I/O action using streams of data on disk or generated from function generators (next section)

d) Allow the user to handle I/O requests

The external world stimulation portion of the file defines a set of value and occurrence generators. This combination causes values to occur at the pins of the simulated device either in response to an I/O request or at a specific time or asynchronously. These generators are especially useful for external interrupt modeling. One should note that if the built-in functionality for external world stimulation isn't sufficient, there are hooks provided for the inclusion of user written routines.

Finally, the recording section of the file tells the simulator which events to record and how. The data can be directed either to a file or to the user's display. The main simulator screen values (Figure 2) are always kept updated in any case.

Remember, all of the foregoing files are generated automatically from the Intel Configuration Utility. The only manual additions would be those of custom external world stimulation routines mentioned above.

Use of the Simulator in the Design Process

With the simulator, the design process changes only slightly. Referring to the bottom timeline in Figure 1, the sequence is:

1. Initial design of the system.
2. Break down system functionality into hardware and software components.
3. Team proceeds with the simultaneous refinement of design details and development of code and hardware.

   At this point the software engineer and the hardware engineer run the ICU to generate the simulated environment for software testing.

Note: *The hardware engineer should be responsible for keeping the simulator in step with the hardware being developed. As changes occur in the hardware, the engineer should notify the software team in addition to changing the ICU file.*

Continued on page 8

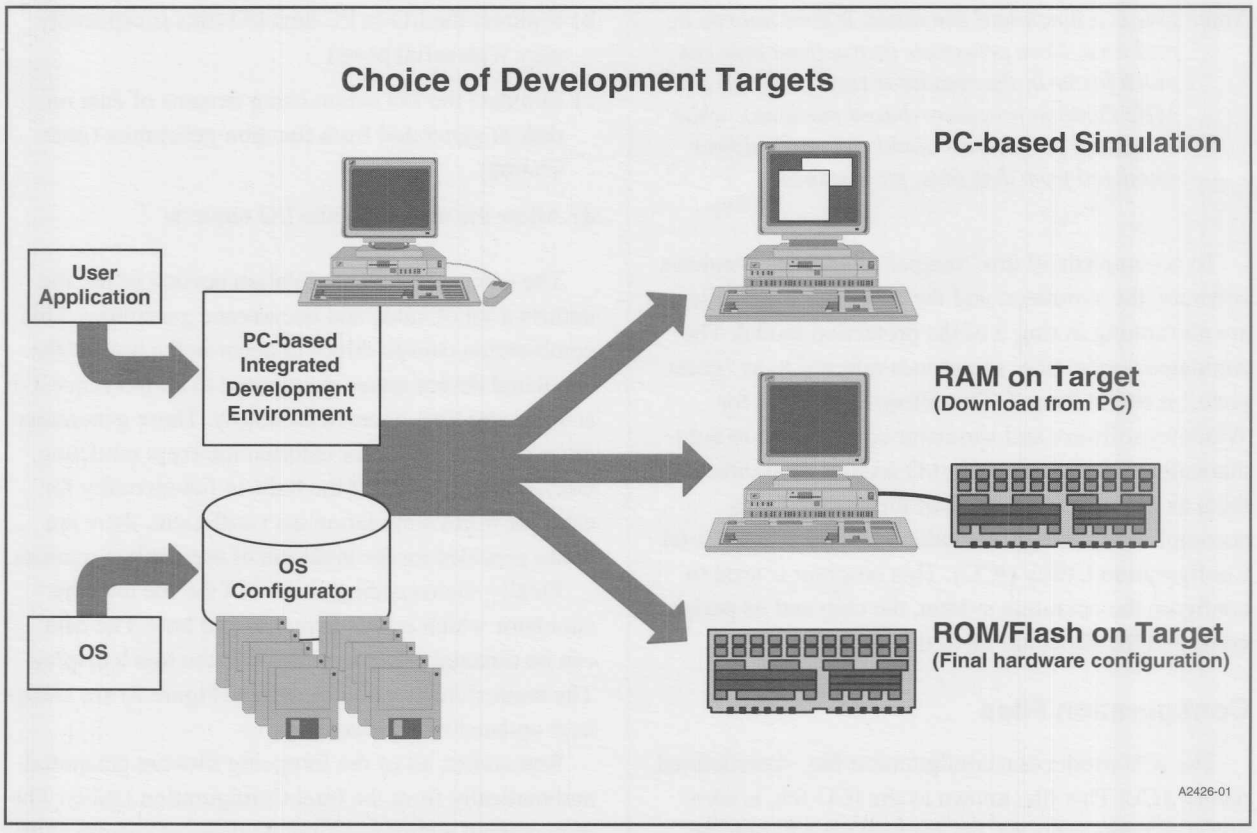

Figure 4. Development Targets

Continued from page 7

4. The software development team tests their code on the simulator/debugger as the hardware development proceeds (upper arrow in Figure 4).
5. When test hardware becomes available, final software testing can proceed using the same debugger on the hardware target through a serial link. If there is a lack of hardware test targets, the simulator/debugger can continue to be used by some members of the team until the project is complete or more hardware is constructed (middle arrow, Figure 4).
6. Changes in hardware and software until final testing is completed satisfactorily.
7. Build of final system; production can begin (lower arrow, Figure 4).

As you can see, the availability of the simulator in the second case eliminates any programming idle time and should reduce final system integration time by providing an accurate test target for the software from the very beginning of testing.

Summary

The use of a simulator/debugger during a project that employs custom hardware can reduce the amount of time necessary for development. The simulator provides a test target from the beginning of testing, sometimes well before actual hardware can be made available. The combination of an ICU and an integrated simulator/debugger allows testing to proceed continuously and almost seamlessly from the simulation phase through the initial hardware target phase to the loading and testing of the final product.

ZapCode — The On-Line ROM Transfer Software Tool

Udo Radlhammer
Business Process Analyst, ECS
Intel Corporation
Article ID #0803

For more information on how to obtain a ZapCode user ID and password, call Intel's ROM Processing Group, in Chandler, AZ, at (602)554-8618.

Intel has come a long way in making it easier for our customers to do ROM business with us. In addition to competitive factors such as pricing and availability, customers have had to deal with a cumbersome and somewhat tedious process of getting the ROM code to Intel. Prior to 1992, customers had the choice of shipping an EPROM, mailing a floppy disk, or sending the ROM code via the ROM Bulletin Board. Some customers sent it directly to Intel's ROM Processing Group in Chandler, Arizona. Others, specifically international customers, sent it to their local sales office, who then sent it to Chandler. At the Chandler site, the code was translated into an 8080 Intelec hex format, which fills the blank ROM area with FFH or 00H. We then returned the reformatted file to the customer using the same method they used to send it to us. Whichever way the customer chose to send their code, it took days or even weeks for some geographic customers to receive the verification file.

Since 1992, U.S. customers have been using "ZapCode," an electronic, masked ROM data submittal service. With this software, customers make one phone call to submit their code and have it returned. The software is based on a UNIX engineering platform and includes toll-free phone access available in the U.S. 24 hours a day. To get a ZapCode user ID, password and ZAPPREP files (ZapCode Preparation program), customers call the ROM Processing Group in Chandler, Arizona. A starter kit is provided to the customer, via a Federal Express shipment, to allow access to the system. ZapCode received rave reviews from customers such as Seagate and Honeywell. During customer visits and surveys, even some international customers such as Novell, Hardie, and Bosch stated that "ZapCode is a great ROM tool and there is no system like it available from the competition."

In June 1994, Intel will introduce an advanced version (Phase II) of ZapCode. This new version will move to a distributable business platform, such as a LAN, rather than an engineering platform. Also, rather than a character-based system, the software will be written in a Windows* software-based Graphical User Interface (GUI) design using Client SQL*Server technology. Since the software will be on Intel's side, rather than the customer's, the installation and service support requirements for the customer will be minimal. Table 1 compares the features of the current (Phase I) and advanced (Phase II) versions.

The new system will support toll-free access for worldwide users and will be accessible 24 hours a day, 7 days a week. It will have an online computer-based training (CBT) tool as well as an online Help function. The transmission time will be reduced because the data entry requirements for the customer will be minimal

Continued on page 10

Table 1. ZapCode Features

Phase I Features	Phase II Features
Toll-free access AMO only	Toll-free access worldwide
20–60 minute transmission time	15–30 minute transmission time
non-Windows	Windows and non-Windows
High cost of service	Lower cost of service
Limited product options	Prompts for full product information
Hard-copy user's manual, classroom training	Online user's manual, computer-based training

Continued on page 10

Continued from page 9

(20 minutes or less!). After a customer's initial usage, the system will retain customer information such as the address and the names of the contact person and end-user of the code. Frequently required information will be incorporated in this version of ZapCode. This includes items such as the customer part marking options, package options, shipment pack requirements, temperature or burn-in requirements, and address range of the ROM array. All of these items are in an easy-to-use, drop-down menu (Figure 1). ZapCode will feature a view option for the part marking, an option to request device security, and a selection for fill value on any blank ROM array. The implementation for this phase of ZapCode is targeted for June 1994.

How ZapCode Phase II Works

Figure 2 provides a high-level view of how the ROM data is transferred to Intel from the customer's PC. To get started, the customer installs the ZapCode Phase II executable file onto a PC. This file will be available from the Bulletin Board System or as a free mailed software kit. Using a 2400 baud or greater Hayes-compatible modem, the customer dials ZapCode via a toll-free telephone number. The system is accessible 24 hours a day, 7 days a week. To access the system, the customer must use a unique ID and password. This ensures that each customer's data is kept secure and cannot be accessed by other users of the system. ZapCode data is transferred twice per day to Intel's production database and is stored there permanently.

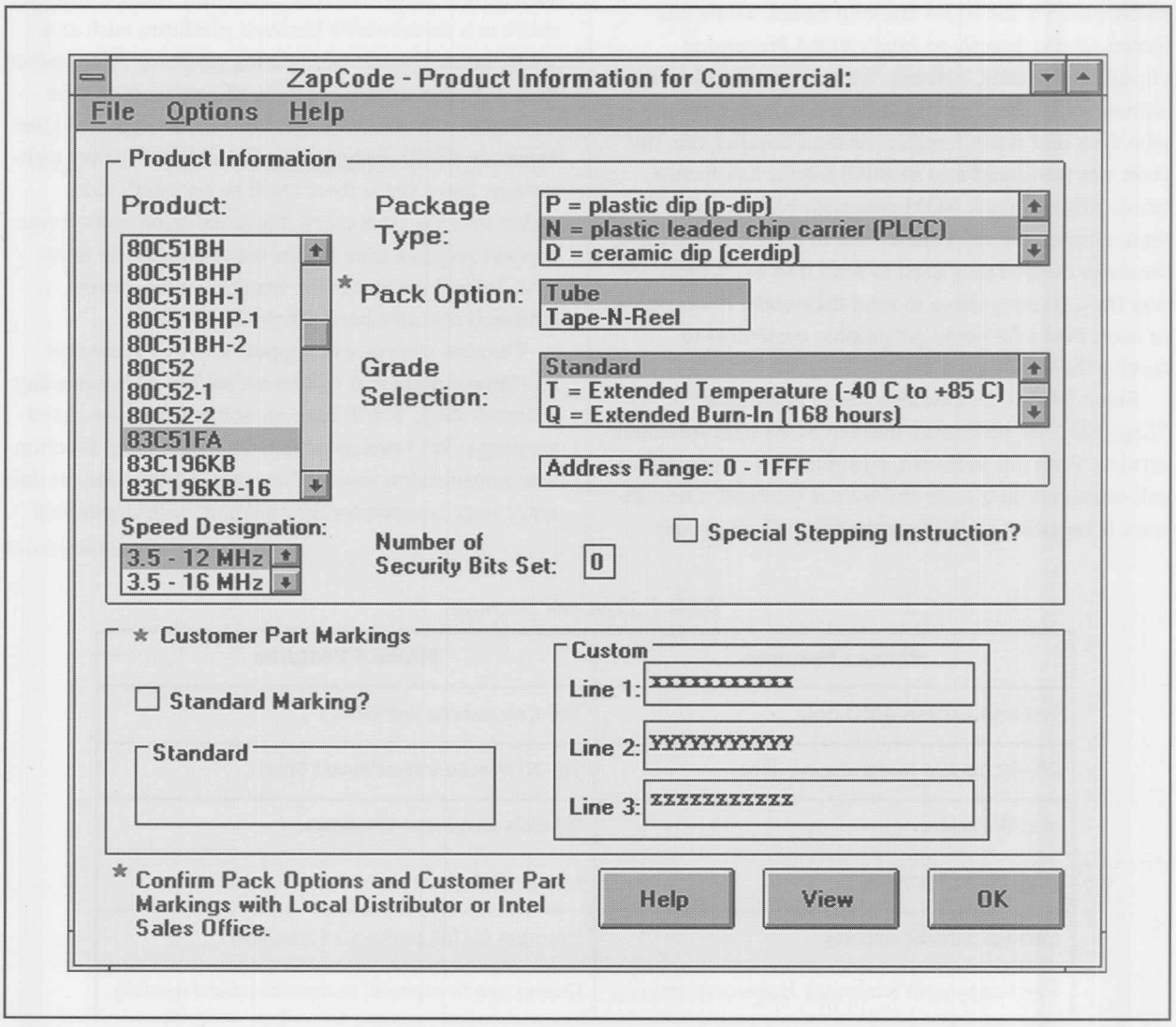

Figure 1. ZapCode Product Information Screen

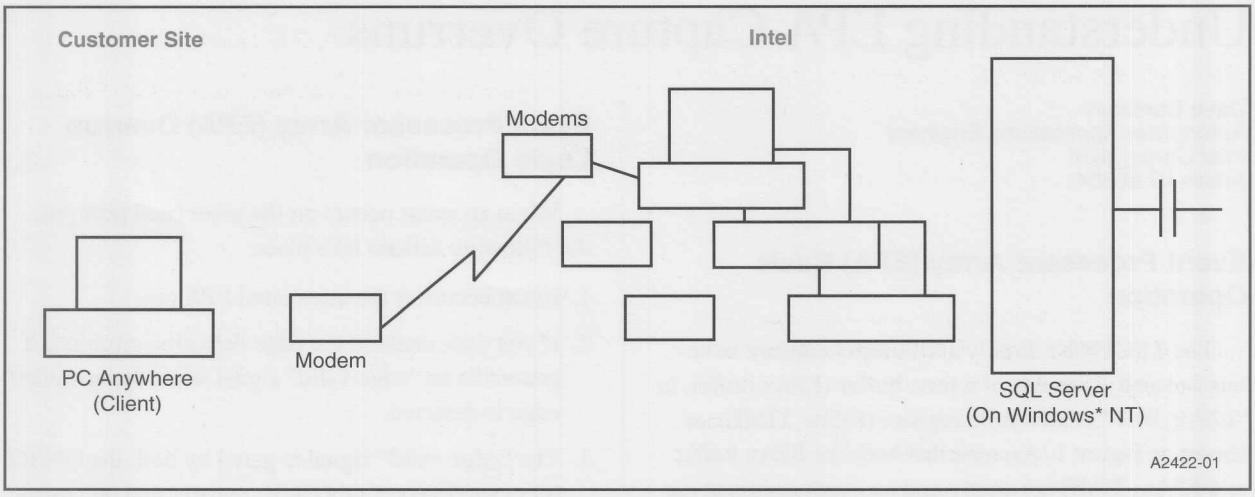

Figure 2. Transferring ROM Data from Customer PC to Intel

ZapCode Phase II Requirements

The ZapCode software requires the following hardware and software configuration:

- IBM AT* compatible personal computer (80386-based or greater with VGA)
- Hayes-compatible modem (2400 baud recommended)
- analog telephone line (tone dial)
- printer
- DOS version 5.0 or greater
- Windows version 3.1 or greater

New Embedded Control FaxBack* Service Documents

New/Updated as of February 21, 1994

Title	Number
Intel Flash Memory Cards and PCMCIA Standard: A Clarification	2097
Press Release: 3 Volt MCS® 51 Microcontroller Press Release	2119
InfoGuide: 3 Volt 8xL51FX (8xL51FA/FB/FC)	2723
MCS 51 Microcontroller: Architecture Development Tools Line Card (includes Product Line Card)	2622
Project BUILDER 196: Applications Proj. Kit and Modeling Software for the 196KB/KC/KD	2643
MCS 96 Microcontroller: Understanding EPA Capture Overruns	2069
EV80960: Eval Board Repair/Return Policy	2620
Article: Low-Voltage MCS 51 Microcontroller Application Story: Pen-Based Computer	2109

Customer Education

Looking for customer training in North America for MCS® 51 or MCS® 96 microcontrollers or i960® microprocessors? The training schedule for the second and third quarter of 1994 includes the following embedded courses:

8051 Microcontroller Family (ED3030)
MCS 96 BH/KB/KC/KD Microcontroller Family (ED3040)
i960 KA/KB/CA Embedded Processors (ED3050)
Applying the Intel386™ EX Microprocessor Architecture (ED3021)

and three **NEW 1994 COURSES:**

Introduction to Fuzzy Logic Technology

Fuzzy Logic Design

Intel i960 P100 Embedded Processor

Most courses are taught in Phoenix, Arizona. To register or to get more information, please call Customer Training at 1-800-234-8806.

Understanding EPA Capture Overruns

Dave Boehmer
Automotive Applications Engineer
Intel Corporation
Article ID #0804

Event Processor Array (EPA) Basic Operation

The 8XC196Kx family's EPA input-capture structure basically consists of a time buffer (EPAx buffer, or "EPAx_BUF") and a time register (EPAx_TIME), as shown in Figure 1. Assume that both the EPAx buffer and EPAx_TIME are empty and an event occurs at the pin. After the event is recognized as valid, the current TIMERx count value will be loaded through the EPAx buffer into EPAx_TIME. The act of data moving from EPAx_BUF to EPAx_TIME is what causes the actual EPA interrupt. When a second event occurs at the pin, the current TIMERx value will be loaded into the EPAx buffer and held there until the user reads EPAx_TIME. A third event would be considered an overrun, since both the time and buffer registers are full. If the ON/RT bit in the EPAx_CONTROL register is a zero, this third event will be ignored. If the ON/RT bit is a one, this third event time will overwrite the second event time in the EPAx buffer. Either way, an overrun interrupt will be generated. When the user reads EPAx_TIME, the contents of the EPAx buffer will automatically be transferred into EPAx_TIME and an EPA interrupt will occur.

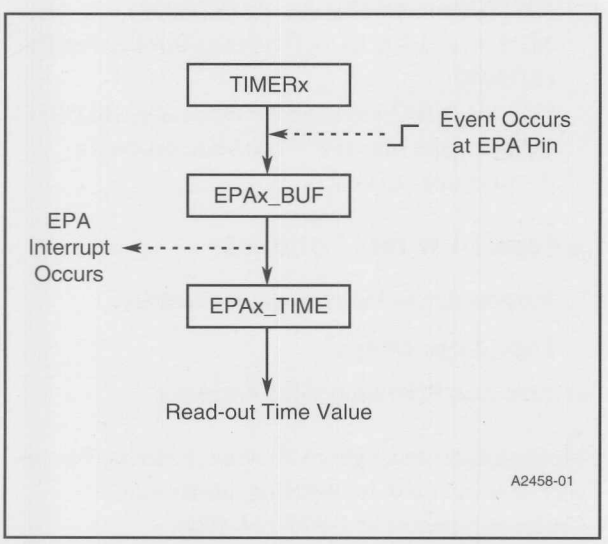

Figure 1. EPA Input-capture Structure

Event Processor Array (EPA) Overrun Logic Operation

When an event occurs on the associated port pin, the following actions take place:

1. Event occurs at the associated EPA pin.
2. If you have enabled the edge detection circuitry, it generates an "edge valid" signal when the specified edge is detected.
3. The "edge valid" signal is gated by both the ON/RT bit of the EPAx_CONTROL register and an "overrun" signal. The "overrun" signal indicates that both the EPAx buffer and EPAx_TIME are full and that an overrun condition will occur. One of four actions will take place at this point:

ON/RT	Buffer/time register status	Action taken when a valid edge occurs
0	empty	Edge is captured and loaded into buffer/time register.
0	full	New data is ignored — no capture, EPA interrupt, or transfer occurs; OVR_INT interrupt occurs.
1	empty	Edge is captured and loaded into buffer/time register.
1	full	Old data is overwritten in the EPAx buffer; OVR_INT interrupt occurs.

4. The "edge valid" signal is used to generate a "capture timer" signal and an "event interrupt/transfer" signal.
   - When activated, the "capture timer" signal initiates transfer of the current TIMERx count value to the EPAx buffer.
   - When activated, the "transfer" signal initiates the transfer of the contents of the EPAx buffer to EPAx_TIME.
   - The "event interrupt" signal sets the corresponding EPA interrupt pending bit.

EPA "Lock-up" Behavior

"Lock-up" is a generic term used to define the case in which the EPA peripheral fails to recognize subsequent edges input on an EPA pin. During lock-up, both the EPAx buffer and the EPAx_TIME register are full without a pending interrupt to signal that a read of EPAx_TIME is necessary. EPA lock-up occurs when a combination of three conditions exist:

- an input signal with a frequency high enough to cause overruns is present on an enabled EPA pin, and
- the ON/RT bit is set (1 = old data is overwritten on overrun), and
- EPAx_TIME is read at the exact instant that the EPA recognizes the captured edge as valid.

The input frequency at which this occurs varies, depending on the length of your interrupt service routine as well as other factors. Unless your code accounts for overruns, an EPA channel that becomes locked up will remain locked up until reset is asserted.

The particular EPA channel will recover from the lock-up condition if the EPAx_TIME register is read. The act of reading EPAx_TIME allows the buffered time value to be moved into EPAx_TIME. This clears the buffer and allows another event to be captured. Remember that the act of the transferring the buffer contents to the EPAx_TIME register is what actually generates the EPAx interrupt.

Preventing EPA Lock-Up

Any one of the following suggestions can be used to prevent or recover from an EPA lock-up.

Set ON/RT to "0"

When the ON/RT bit is zero, the EPA does not consider the captured edge until the EPAx_TIME register is read and the data in the EPAx buffer is transferred to EPAx_TIME. This prevents lock-up by ignoring new input capture events when both the EPAx buffer and EPAx_TIME contain valid capture times.

Since the ON/RT bit gates the "edge valid" signal, the "capture timer" and "event interrupt/transfer" signals are never generated. The "edge valid" signal will still be active to indicate that an overrun occurred. The OVR_INT pending bit in EPA_PEND is set along with the EPA_INTx bit in the INT_PEND register. At the same time, the data from the EPAx buffer is transferred to EPAx_TIME and another EPAx interrupt is generated. This means that a lock-up never has the opportunity to occur. In this situation, the OVR_INT will always be pending, waiting until the EPAx interrupts are serviced, because the EPA_INT always has higher priority.

Enable the OVR_INTx interrupt and read the EPAx_TIME register within the ISR

This is more a reactive method for resolving the lock-up condition as opposed to a proactive method for preventing it. Using this method, you handle overruns within your OVR_INTx ISR (interrupt service routine) in a way that is acceptable for your particular application. When a lock-up occurs, EPAx interrupts will be locked up, but the "overrun" signal will be generated to set the pending bits in the EPA_PEND register. The OVR_INTx interrupt will then be acknowledged and its interrupt service routine will read EPAx_TIME. The read of EPAx_TIME will generate an EPAx pending interrupt and clear the overrun condition. This will cause the device to come out of the lock-up.

Check for pending EPAx interrupts before exiting an EPAx ISR.

Another method for avoiding a possible lock-up is to check for pending EPAx interrupts before exiting an EPAx ISR. This is an easy way to detect overruns and additional interrupts. It can also save loop time by eliminating the latency necessary to service the pending interrupt. However, this method cannot be used with the Peripheral Transaction Server (PTS), since your code cannot service additional interrupts during a PTS cycle. If your system uses the PTS, you should choose one of the other methods.

Summary

An EPA channel can lock up when events occur at an EPA input pin faster than a given EPAx ISR can process them unless a methodology is in place to address the overruns. Three solutions are presented to deal with EPA input overruns. If EPA overruns are possible, it is important to either ignore them (set ON/RT to zero) or address them in an interrupt service routine. Regardless of the method used for preventing EPA lock-up, it is important to comprehend the maximum input frequencies you may encounter in a given application as well as the ISR execution time.

Understanding Program Security on MCS® 51 Microcontrollers

David Elting
Applications Engineer
Intel Corporation
Article ID #0805

Introduction

Increasing interest in program security has identified the need to clarify the security features of MCS® 51 microcontrollers and the methods required to use these features. These devices offer two possible security features: security lock bits and encryption arrays. Various products offer none, one, or both of these features. Table 1 lists the available automotive MCS 51 microcontrollers and their security features, to help eliminate some of the confusion about security. The sections following the table describe these security features and explain how to use them.

Table 1. Security Features of Automotive Products

Product Name	Security Lock Bits	Encryption Array
8031AH	N/A	N/A
8032AH	N/A	N/A
8051AH	N/A	N/A
8051AHP	1 Bit *	N/A
8052AH	N/A	N/A
80C31BH	N/A	N/A
80C51BH	N/A	N/A
80C51BHP	1 Bit *	N/A
87C51	2 Bits	32 Bytes
87C54	3 Bits	64 Bytes
80C51FA	N/A	N/A
83C51FA	N/A	N/A
87C51FA	3 Bits	64 Bytes
87C51FB	3 Bits	64 Bytes
87C51FC	3 Bits	64 Bytes

**The security lock bit for the 8051AHP and 80C51BHP is embedded in the ROM mask. This security feature is always set on these devices.*

Security Mechanisms

The first security mechanism consists of one or more security lock bits. Programming the security lock bits limits access to the ROM contents. Table 2 details the protection schemes offered by the lock bits. You can program the security bits to achieve the desired protection level. Remember that once a security bit is programmed, it cannot be erased without erasing the entire EPROM. This makes dynamic failure analysis impossible. Therefore, Intel cannot accept devices returned for failure analysis if you have programmed the lock bit(s).

The second protection mechanism available on various devices is the encryption array. The encryption array logically Ex-NORs the internal ROM contents. This in effect scrambles the contents of internal memory, which hampers the ability to decode the ROM without the key. One quick reminder when utilizing the encryption array: if a code byte contains the value 0FFH, verifying the code byte will produce the encryption byte value. Therefore, a large block (greater than 64 bytes) of code memory left unprogrammed will display the encryption array contents during a read-out. Consequently, if you use the encryption array, you should consider programming all unused internal code bytes with random values other than 0FFH, preferably changing the value, to ensure maximum program protection.

The location of the security lock bits and encryption array on MCS 51 microcontrollers is concealed within the EPROM rather than existing at a given memory location. Special programming modes access these hidden memory locations when programming the security bits and encryption array. However, erasing the EPROM also erases the contents of the encryption array and security bits, allowing reprogramming of the EPROM.

Securing an MCS® 51 Microcontroller

Securing an MCS 51 microcontroller requires a few simple techniques:

1. Ensure that the device you are using offers the desired security features.
2. Determine whether your EPROM programmer supports programming the security features.

Table 2. Lock Bit Protection

	Lock Bits			Protection Description
	LB1	LB2	LB3	
1	U	U	U	No Program Lock features enabled. (Code verify will still be encrypted by the Encryption Array, if programmed).
2	P	U	U	MOVC Instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on Reset, and further programming of the EPROM is disabled.
3	P	P	U	Same as 2, plus verification is disabled.
4	P	P	P	Same as 3, plus external execution is disabled.

U = Unprogrammed, P = Programmed

3. If the device supports an encryption array, determine an effective encryption code consistent with the encryption array size.
4. If the device supports security lock bits, determine the protection scheme best suited for the desired level of code security.
5. Program the device with the security features as supported by the programmer. For more information on programming lock bits, please refer to the Intel *Embedded Microcontrollers and Processors* handbook (order number 270646) and the programming manual that accompanied your EPROM programmer.

ROM Code Submittal for Devices With Security Features

When you submit a ROM code for a device that supports security features, Intel will enable security only if you have specified it on the ROM Code Order Form. However, because of testability issues, Intel imposes some guidelines for ROM code submittal. The following paragraphs clarify the ROM code process as it relates to encryption arrays and security bits.

The most important thing to remember when submitting a ROM code that implements security features is that Intel will program only the first lock bit. To ensure code security, Intel always programs an encryption array in conjunction with the lock bit. Because of the protection associated with the remaining lock bits, Intel cannot support proper testability of the device once the second or third lock bit is programmed. If you need the added levels of security, you must program the remaining security lock bits once you are confident that the device has been fully tested.

To permit programming of the encryption array, you must submit an encryption array file along with the program hex file. The encryption array file is an independent hex file apart from the program hex file. When filling out the ROM Code Order Form, type the name of the encryption file on the "Encryption File Labeled" line. Do not type the encryption code on this line.

Programmer Support Information

Not all EPROM programmers support encryption programming or de-encrypting the contents of memory. Check with the programmer manufacturer to see whether the programmer you are using supports these features.

Summary

MCS 51 microcontrollers provide various levels of program security. You can choose a device that supports the level of security that your application requires. You can either program the security features yourself or provide an encryption array file and have Intel enable the features for you. Intel can program only the first security lock bit, but the encryption array helps to ensure code protection. To achieve a higher level of protection, you can program additional lock bits after the device is fully tested.

Jennie Abella
Applications Engineer
Intel Corporation
Article ID# 0806

Introduction

The Auto Programming mode is a low-cost programming alternative. Using this mode, the contents of an external EPROM are copied to the internal OTPROM of the 87C196KD. This article explains how to construct an inexpensive programmer for the 87C196KD. A programmer for the 27512 is still necessary, but these are readily available. The programming circuit in this article is recommended for programming the 87C196KD and can also be used for the 87C196KC.

Auto Programming Algorithm

The 87C196KD enters a programming mode when V_{PP} (typically 12.5V) is applied to EA# during the rising edge of RESET#. Then the following sequence is initiated:

1. The upper port 0 pins P0.4–7 (PMODE pins) are read to determine which programming mode is selected. If the value is 1100 (binary), the auto programming mode is entered.
2. The internal OTPROM location 2018H (CCR security lock bits) is checked to see if the security key has been programmed. If these bits are programmed (=0), then the internal security key locations (2020H–202FH) are checked with external locations E020H–E02FH. If security is passed, the program continues. If not, the device enters an endless loop and must be reset to exit it. The LED to indicate that programming has started will not light. (Follow the powerdown procedure and then another attempt can be made to program the device.)
3. The PPW is then loaded from external locations 2014H/2015H, which sets up a 100µs programming pulse. (The equation for this PPW value is shown in "Calculating the Programming Pulse Width" in this article.)
4. PACT# is activated (P2.7=low) to indicate that programming has started. (LED will light.)
5. PVER is initialized (P2.0=high) to indicate that there are no errors to this point. (LED will not be lit.)
6. External data is then read, starting at 4000H.
7. If the word is not 0FFFFH, then the Modified QuickPulse subroutine is called (Step 8). If the word is 0FFFFH, then the word is not programmed and the address is checked to see if programming is completed (Step 10).
8. The PPW timer is started and the data word is written to the OTPROM. The program waits until an interrupt is caused by the PPW timer, which ends the programming pulse. If five writes have not been written, this step is repeated. When five writes have been written to this location, then programming continues with the next word.
9. This location is then read back from the OTPROM and compared to the external location. If it did not program correctly, then PVER is deasserted (P2.0=low) and the LED lights, indicating an error. **Programming will continue even if an error is detected**. The part cannot be programmed again.
10. The address is checked to see if programming has been completed. If the address is 0BFFEH, programming is completed and PACT is deactivated (P2.7=high) and an infinite loop is entered. Continue with the power-down procedure. If programming is not completed, the address pointer is incremented and the next word is programmed.

Table 1. Auto Programming Memory Map

External EPROM Address	Internal OTPROM Address	Description
2014H	N/A	PPW Least-Significant Bit
2015H	N/A	PPW Most-Significant Bit
4000H–BFFFH	2000H–9FFFH	Reserved locations for code and data.
E020H–E02FH	2020H–202FH	Security key, during verification.

Auto Programming Circuit

Figure 1 is the recommended circuit for auto programming the 87C196KD. Since BUSWIDTH is low, an 8-bit external bus is used. Code to be programmed in the KD at 2000H must be located in the external EPROM starting at location 4000H. The memory remapping is shown in Table 1. The PPW must be located in the external EPROM at location 2014H and 2015H. (See "Calculating the Programming Pulse Width" in this article.)

P0.4–P0.7 are hardwired to V_{SS} and V_{CC} and determine the programming mode. P1.0–P1.2 are used to generate the upper bits of the external EPROM address. The status outputs PACT# and PVER are buffered by a 74HC14 and drive LEDs to indicate Programming ACTive (PACT#) and Programming VERification (PVER). All unused inputs are connected to ground (V_{SS}) and unused outputs are left floating. READY, NMI, and BUSWIDTH are active and should be connected as indicated.

Auto programming is specified for a crystal frequency of 6 to 8 MHz. A 27(C)512 EPROM with tACC = 250ns and tOE = 100ns or faster specifications should be used.

Power-Up and Power-Down Sequencing

Important: To avoid damaging the 87C196KD during auto programming, follow these rules:

Power-Up Sequence

1. When first powering up, the device must be held in reset while V_{CC} stabilizes. V_{PP} and EA# must be allowed to float during this time.
2. Continue holding the device in reset after V_{CC} has stabilized, and apply +12.5 volts to EA# and V_{PP}. Refer to the data sheet for exact specifications on this voltage.

   Warning: Applying voltage to V_{PP} when V_{CC} is low will permanently damage the device. The recommended circuit shows a switch that interlocks the V_{PP} switch to help prevent damage to the device.
3. Allow time for EA# and V_{PP} to be within tolerance and for the oscillator to stabilize.
4. After condition 3 is met, RESET# may be allowed to rise.

Continued on page 18

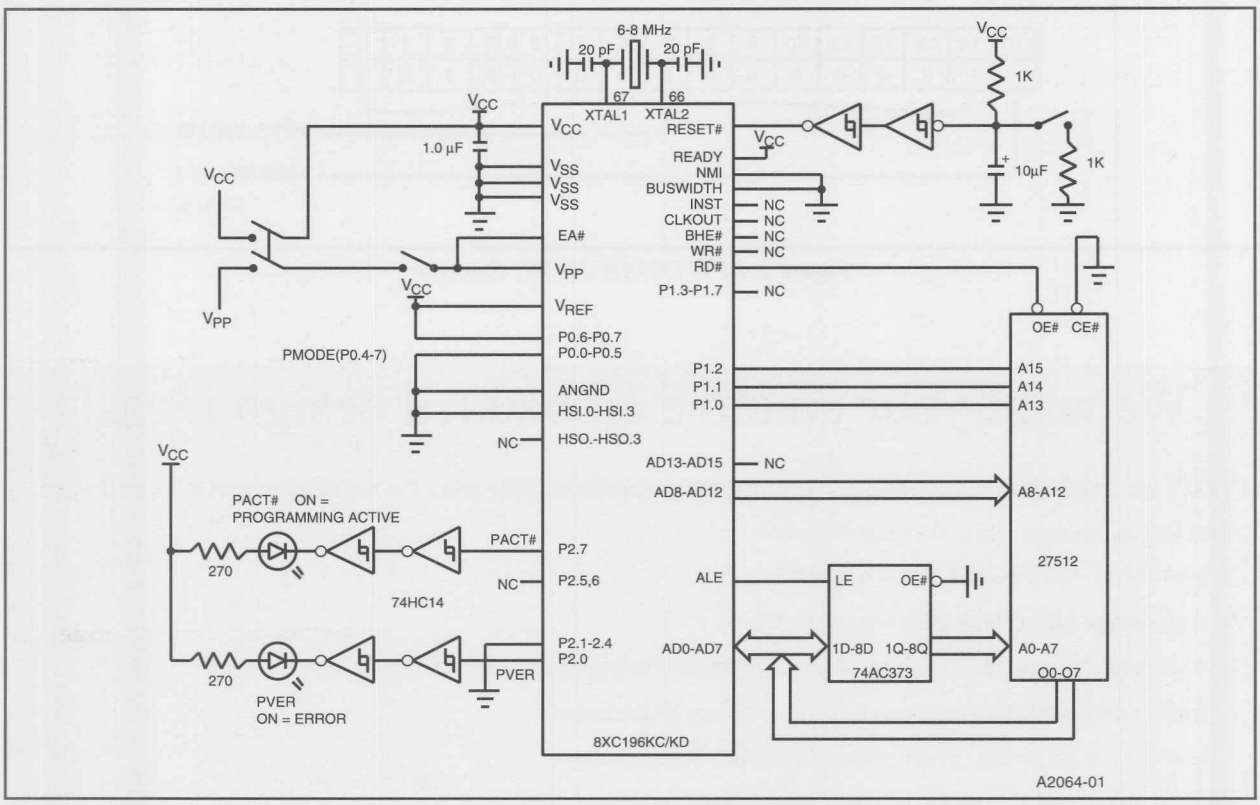

Figure 1. Auto Programming Mode Circuit

Continued from page 17

5. As soon as RESET# rises, the auto programming sequence begins and the PACT# LED turns on.
6. After completion of the auto programming sequence, the PACT# LED turns off and the power-down sequence should be followed.

Power-Down Sequence

1. Assert the RESET# signal (RESET#=0). RESET# must be held low throughout the powerdown sequence. Do not allow the reset signal to "bounce," or another programming sequence will begin.
2. Remove the +12.5 volts applied to EA# and V_{PP} and allow these pins to float.

   Warning: EA# and V_{PP} must be allowed to float before removing V_{CC} or the device will be damaged.
3. Turn off the V_{CC} supply and allow time for this to reach 0 volts.
4. The device can now be removed from the auto programming circuit.

Calculating the Programming Pulse Width

The programming pulse width needs to be 100µs for the device to program correctly. Use the following formula to calculate the PPW_VALUE. Round the PPW_VALUE to the next higher integer value and load this value in the PPW location in the external EPROM.

$$PPW\_VALUE = (0.6944 \times XTAL1) - 1$$

Where PPW_VALUE is a 15-bit word and XTAL1 is the crystal frequency in MHz.

For example, with an oscillator of 8MHz,

$$PPW\_VALUE = (0.6944 \times 8) - 1$$
$$= 5.55 - 1$$
$$= 4.55$$

This would be rounded to 5. The MSB must always equal 1 (see Figure 2); therefore, for this example, location 2014H/2015H = 8005H.

Note: *External EPROM location 2014H is loaded with the LSB and 2015H is loaded with the MSB of PPW_VALUE.*

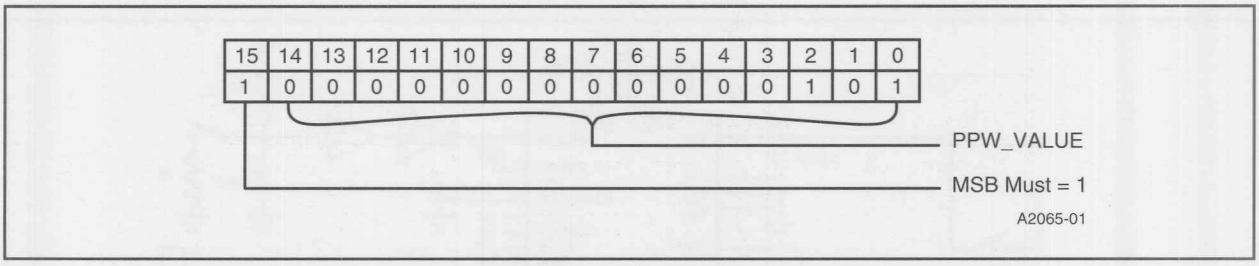

Figure 2. PPW (2014H/2015H) Example

The Embedded Solutions Seminar II (ESS II)

City	City	Date
Seattle, WA Marriott Sea-Tec Airport	San Diego, CA Marriott Mision Valley	Tuesday, April 12th
	Orange County, CA Anaheim Hilton	Wednesday, April 13th
Santa Clara, CA Techmart	Sherman Oaks, CA University Hilton & Tower	Thursday, April 14th
Santa Clara, CA Techmart		Friday, April 15th
Baltimore, MD Sheraton, BWI	Orlando, FL Embassy Suites South	Tuesday, April 19th
King of Prussia, PA Hilton Valley Forge	Duluth, GA Gwinnett Civic Center	Wednesday, April 20th
Boxborough, MA Host Hotel	Mayfield Heights, OH Landerhaven Conference Ctr	Thursday, April 21st
Denver, CO Marriott Southeast	Bloomington, MN Marriott Bloomington	Tuesday, April 26th
Austin, TX Austin Hilton	Westmont, IL Inland Conference Center	Wednesday, April 27th
Dallas, TX Westin Galleria	Mississauga, Ontario Canada Toronto Airport Hilton	Thursday, April 28th
Milan, Italy Milan Milanofiori	Zürich, Switzerland Regensdorf Môvenpick	Tuesday, May 3
Paris, France Meridien Etoile	Vienna, Austria Vienna Plaza	Wednesday, May 4th
Brussels, Belgium President	Munich, Germany Ramada Freising	Thursday, May 5th
Rotterdam, Holland World Trade Center	Düsseldorf, Germany Hilton	Friday, May 6th
Stockholm, Sweden Berns	Madrid, Spain Hussa Princesa	Monday, May 9th
	London, England Kempton Park	Tuesday, May 10th
Prague, Czech Forum	Manchester, England Britannia	Wednesday, May 11th
Warsaw, Poland Marriott		Friday, May 13th
	Tel Aviv, Israel Dan Accadia	Thursday, May 19th
Osaka, Japan		Wednesday, May 25th
Tokyo, Japan		Thursday, May 26th

To register, call 1-800-368-4110 in North America or +44(0)793-431155 in Europe.

ERRATA AND CHANGE IDENTIFIERS

Article ID #0807

Change identifiers have been used since 1990 to distinguish revisions, or steppings, of embedded control devices. Older devices have no change identifiers. On most devices, the change identifier is the last character in the FPO number, which is typically a nine-character code on the second line on the top of the device. An example FPO number is "L1234567D," in which "D" is the change identifier. On some devices, such as the 8XC51SL-BG, the change identifier is a separate line item and uses several characters. For example, change identifier "SW011" identifies the B-3 stepping of the 8XC51SL-BG.

This article lists change identifiers, errata, and design considerations for recent steppings of embedded control products. For many of these devices, complete errata listings or explanations are available from the FaxBack* service. The "Ref" column in each table lists FaxBack service document numbers for errata lists and explanations, and "For More Information" at the end of this article has a complete list of related document numbers and titles.

MCS® 51 Microcontroller Family Errata and Design Considerations

This list covers the most recent steppings of the MCS® 51 microcontrollers. A complete list of the errata for all steppings is available on the FaxBack service (document #2632).

Table 1. MCS® 51 Microcontroller Family Errata and Design Considerations

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
8051AH/8031AH	C	A	1. External interrupt 0 errata	2154 2161
	C-3	B	No known errata	
80C51BH/80C31BH	C	none	1. Reset lockup problem 2. High IPD if C does not equal B.7 before going into powerdown 3. ROM verify mode fails 4. Steam passivation problem on plastic parts	
	C-1	none	1. High IPD if C does not equal B.7 before going into powerdown 2. ROM verify mode fails	
	D	D or 2	No known errata	
87C51	D	A	No known errata	2106
80C52/80C32	C	none	1. RST/ONCE mode problem	
	A	A	No known errata	
83C51FA/80C51FA	C	none	1. PCA errata	2528
87C51FA	C	none	1. RST/ONCE mode problem 2. PCA errata	2528
	D	A	No known errata	2107
8XC51FB	A	none	1. PCA errata	2528
	B	A	No known errata	2111

Table 1. MCS® 51 Microcontroller Family Errata and Design Considerations (Continued)

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
8XC51FC	A	none	1. Port 1, 2, 3 problem — asychronous port reset not supported 2. Failed ESD qual testing	2028
	B	none	No known errata	
8XC51GB	B	none	1. Reset polarity changed to active low 2. Port 1 reset value changed to all zeros	2032 2032
	B-2	none	No known errata	
8XC152JX	B	none	1. AE/RDN race condition 2. Receive FIFO is not cleared when receiver is enabled 3. DMA errata 4. SDLC flag recognition errata	 2030 2118 2035
	C	none	No known errata	
8XC51SL-BG	B-3	SW011	1. GATEA20, RCL hardware speedup processing 2. Powerdown current stabilization 3. Port 2 address mux 4. KSI powerdown wakeup interrupt 5. Reset errata	2008 2114
	B-4	SW062	1. GATEA20, RCL hardware speedup processing 2. Powerdown current stabilization 3. Port 2 address mux 4. KSI powerdown wakeup interrupt	2008
8XC51SL-AH/AL	A-0 A-1 A-2	AA BA CA	1. Power-down current stabilization 2. System power management errata	2048

MCS® 96 Microcontroller Family Errata and Design Considerations

This list covers the most recent steppings of the MCS® 96 microcontroller. Complete lists of the errata for all steppings are available on the FaxBack service for the 8X9XBH (#2134), the 8XC196KB (#2548), the 8XC196KC (#2136), the 8XC196KR (#2527), and the 8XC196NT (#2178).

Table 2. MCS® 96 Microcontroller Family Errata and Design Considerations

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
8X9XBH	D	D	1. Indexed 3-operand multiply 2. HSI FIFO 3. Reserved location 2019H 4. RESET and the QBD pins 5. Software RESET timing 6. Using T2CLK for Timer2	2134
	E	E	1. Indexed 3-operand multiply 2. HSI resolution 3. Reserved location 2019H 4. Reserved location 201CH	 2631 2140

Table 2. MCS® 96 Microcontroller Family Errata and Design Considerations (Continued)

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
8XC196KB 8XC196KB10/KB12	B	B	1. Divide during HOLD or READY 2. HSI 8/9 state 3. SIO framing error 4. SIO RI flag 5. DJNZW instruction 6. CMPL with R0 7. ALE glitch	2548 2156 2568
8XC196KB/KB16	B	D	1. Divide during HOLD or READY 2. HSI 8/9 state 3. CMPL with R0 4. Missed EXTINT P0.7	2122 2156 2049
	C	E	1. HSI 8/9 state 2. CMPL with R0 3. Missed EXTINT P0.7	 2156 2049
8XC196KC			The number following each entry in this list is a cross-reference to the applicable section of FaxBack service document #2136.	2136
	all		Design Considerations 1. Indirect shift count value 116 2. Write cycle during Reset 147	
	A	A or none	**Errata** 1. A/D convert error 101 2. BMOVI instruction 105 3. Divide error during hold 109 4. HSO IOC1 bits interchanged 115 5. Port 0 latched on wrong phase 126 6. PTS Req during INT latency 131 7. NMI during PTS latency 132 8. SIO Mode 0 140 9. TIJMP INDEX_MASK value 143 10. Serial Port Framing Error 150 11. QBD glitch during powerup 163 12. A/D linearity too large 173 13. Analog input latch-up 174 14. INST weak during CCB fetch 175 15. 2 CCB fetches 176 16. 3 wait states during CCB 177 17. Pullups too weak during Reset 178 18. Buswidth always 16 bits 179 19. Vcc glitch resets device 180 20. Incomplete reset at > 12 MHz 181 21. Oscillator startup 215 22. Reset hysteresis 216 23. Missed EXTINT P0.7 2049	
	B-1	B	1. Divide error during hold 109 2. NMI during PTS skips address 123 3. QBD glitch during powerup 163 4. ONCE mode entry 214 5. Oscillator startup 215 6. Reset hysteresis 216 7. Missed EXTINT P0.7	 2049
	B-3	D or E	1. Divide error during hold 109	

Table 2. MCS® 96 Microcontroller Family Errata and Design Considerations (Continued)

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
8XC196KC (Continued)			2. NMI during PTS skips address 123 3. QBD glitch during powerup 163 4. Reset hysteresis 216 5. Missed EXTINT P0.7	2049
8XC196KD	A-1	B	1. Missed EXTINT P0.7	2049
8XC196KR/JR/KQ/JQ	A, C	A, C	Design Considerations 1. P6_REG not updated immediately 2. Write cycle during reset 3. EPA timer reset/write conflict 4. Valid time matches 5. CLKOUT 6. Indirect shift operation 7. Internal RAM powerdown leakage 8. A/D latchup 9. INST operation 10. KQ/JQ memory map	2527
	A	A	**Errata** 1. Oscillator noise sensitivity 2. Slave programming mode 3. A/D abort 4. PTS with other interrupts 5. PTS/NMI conflict 6. Data output register cleared when mode register is written 7. Divide error during Hold/Ready 8. SIO Mode 0 9. Remap mode on EPA3 10. Serial port framing error 11. EPAIPV value multiplied by 2 12. EPA_MASK1/EPA_PEND1 must be written as words 13. Interruptable block move (BMOVI)	
	A, C	A, C	1. Ioh2 = – 6 µA	
	C,D	C,D	1. Cannot access external locations 1B00H-1BDFH	
8XC196NT	D	D	Design Considerations In bus controller modes 1 and 2, in 8-bit bus mode, the upper address lines need to be latched	
8XC196MC/MD	B	B	No known errata	
8XC196NP	A	A	1. Illegal Opcode interrupt vector not taken.	

8XC186/8XC188 Family Errata and Design Considerations

Table 3. 8XC186/8XC188 Family Errata and Design Considerations

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
80C186A, B	none		1. Non-contiguous Interrupt Acknowledge cycles 2. ERROR# processing during FWAIT instructions 3. Input high voltage requirement on SRDY and ARDY pins 4. Interrupt Status Register (DHLT and Timer Interrupts) 5. Bus preemption errata (HOLD/HLDA protocol) 6. 80C188 RFSH# pin output timing	2096
80C186XL	A	none	Never put into production	
	B	A	1. INTx/INTAx in Cascade Mode	2025
	C	B	No known errata	
80C186EA/80L186EA	A	A	1. Low hysteresis on RESIN# pin 2. TEST/BUSY#, RD#/QSMD#, LCS#, and UCS# input low voltage 3. INTx/INTAx in Cascade Mode	2025
	B	B	1. INTx/INTAx in Cascade Mode	2025
80C186EB/80L186EB	A	A or none	1. Entry into ONCE mode 2. Low hysteresis on RESIN# pin 3. SINT1 input not latched internally 4. Ready input during INTA# bus cycle 5. CLKOUT transitions on the rising of CLKIN instead of the falling edge 6. I/O ports 1 and 2 initialize to Port instead of Peripheral function (documentation error) 7. INTx/INTAx in Cascade mode	2025
	B	B	1. INTx/INTAx in Cascade Mode	2025
80C186EC	A	A	1. Early exit from Reset (with high Vcc, or low temperature, or both) 2. Powersave Mode initialization at Reset	
	B	B	No known errata	

For More Information

The following FaxBack* service documents contain errata lists and explanations.

Title	Number
MCS® 51 Controller Errata	
MCS 51 Controller: Errata and Design Considerations	2632
MCS 51 Controller: 1991 8 bit Microcontroller Book Errata	2123
MCS 51 Controller: 1992 8-bit Data sheet Errata	2165
8XC51FX: QFP Pin 39 Change	2124
8XC51FX: PCA / TIMER2 Errata	2528
8XC51FX: Hardware Description Errata	2017
87C51FA: D Step	2107
87C51FB/83C51FB: B Step	2111
87C51FC: Data Sheet Errata I	2024
87C51FC: Data Sheet Errata II	2020
87C51: D Step	2106
87C54/80C54: Data Sheet Errata I	2022
87C54/80C54: Data Sheet Errata II	2021
8051/31AH: Shrink C-Step External Interrupt 0 Errata	2161
80C31/51BH: D-Step Marking	2015
87C51GB: A-1 & B Errata & Design Consider	2032
8XC51SLAH/AL Errata	2048
80C51SL-BG: Errata Version 2.6	2008
80C51SL-BG: Product Preview Data Sheet Errata	2630
80C51SL-BG: Errata	2130
80C51SL-BG: Reset Errata	2144
80C152: SDLC Flag Recognition Bug	2035
8XC152: External Demand DMA Bug	2129
8XC152: Global Serial Channel Bug	2030
8XC152: DMA Bug	2118
8XC152: Errata and Clarifications III	2043
MCS® 96 Controller Errata	
8XC196KB/KC/KD,8XC198,8XC194: HSI Events (9 or greater)	2052
8XC196KB: History and Errata	2548
8XC196KB and 8XC198: Bug List	2135
8XC196KB: ALE Glitch	2568
8XC196KB/KC/KD: Missed EXTINT Interrupt Problems on PO.7	2049
8XC196KC: 1991 Handbook Errata	2131
8XC196KC: HSI PTS Handbook Errata	2141
8XC196KC: Bug List	2136
8XC196KC/KD: 1992 User's Manual Errata	2570
8XC196KR/JR/KQ/JQ: Errata	2527

Title	Number
MCS® 96 Controller Errata (Continued)	
8XC196: Hold/Ready DIV/DIVB Problem	2122
8XC198 and 8XC196KB/KB16: Data sheet Errata	2569
8X9XBH, 8X9XJF: Bug List	2134
8X9X: Reading 201CH Bug	2140
186 Errata	
80C186: C186/188 Compatibility with 80C186XL / C188XL C-Step	2132
80C186: AMD 80C18x to INTEL 80C186/186XL Comparison	2540
80C18x XL/EA/EB: INTx/INTAx# Errata	2025
EV80C186EB: Eval Board Manual Errata	2158
EV80C186: 186 Family Eval Board Errata	2592
186XL/EA/EB/EC: 186 Family User's Manual Errata	2603
80C186/C188: B Step Technical Bulletin	2100
80C186/C188XL: B-Step Technical Bulletin	2101
80C186/C188EA: A-Step Technical Bulletin	2102
80C186/C188EB: A-Step Technical Bulletin	2103
80C186/C188EB: B-Step Technical Bulletin	2104
80C186/C188EC: A-Step Technical Bulletin	2105
80C18xEC/80L18xEC: B-Step Technical Bulletin	2057

Sales Offices and Distributor Addresses

Intel Corporation
Literature Department
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
United States

Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi
Ibaraki, 300-26
Japan

Intel Corporation S.A.R.L.
1, Rue Edison, BP 303
78054 Saint-Quentin-en-Yvelines Cedex
France

Intel Corporation (U.K.) Ltd.
Pipers Way
Swindon
Wiltshire, England SN3 1RJ
United Kingdom

Intel GmbH
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen
Germany

Intel Semiconductor Ltd.
32/F Two Pacific Place
88 Queensway, Central
Hong Kong

Intel Semiconductor of Canada, Ltd.
190 Attwell Drive, Suite 500
Rexdale, Ontario M9W 6H8
Canada

U.S. INTEL SALES OFFICES — ***Call (800) 628-8686 to contact any of the following U.S. sales offices:***

Alabama, Huntsville; **Arizona,** Phoenix; **California,** Roseville, San Diego, San Jose, Santa Ana, Sherman Oaks; **Colorado,** Denver; **Connecticut,** Danbury; **Florida,** Deerfield Beach, Maitland; **Georgia,** Norcross; **Illinois,** Schaumburg; **Indiana,** Indianapolis; **Maryland,** Annapolis Junction; **Massachusetts,** Westford; **Michigan,** West Bloomfield; **Minnesota,** Bloomington; **New Jersey,** Red Bank; **New York,** Fairport, Fishkill, Islandia; **Ohio,** Beachwood, Dayton; **Oklahoma,** Oklahoma City; **Oregon,** Beaverton; **Pennsylvania,** Blue Bell; **South Carolina,** Columbia, Greenville; **Texas,** Austin, Dallas, Houston; **Utah,** Murray; **Washington,** Bellevue, Spokane; **Wisconsin,** Brookfield

CANADIAN SALES OFFICES — ***Call (800) 628-8686 to contact any of the following Canadian sales offices:***

British Columbia, Intel Semiconductor of Canada, Ltd., Vancouver
Ontario, Intel Semiconductor of Canada, Ltd., Ottawa; Intel Semiconductor of Canada, Ltd., Rexdale
Quebec, Intel Semiconductor of Canada, Ltd., Pt. Claire

U.S. DISTRIBUTORS:

Alliance Electronics, Inc. / Almac/Arrow Electronics / Arrow Commercial Systems Group / Arrow/Schweber Electronics / Avnet Computer / Hamilton/Avnet / MTI Systems / MTI Systems Sales / North Atlantic Industries Systems Division / Pioneer-Standard / Pioneer Technologies Group / Wyle Laboratories

CANADIAN DISTRIBUTORS:

Almac-Arrow Electronics / Arrow Commercial Systems Group / Arrow/Schweber Electronics / Avnet Computer / Hamilton/Avnet / Zentronics

EUROPEAN INTEL SALES OFFICES:

Finland, Helsinki, (358) 0 544 644
France, St. Quentin-en-Yvelines Cedex, (33) (1) 30 57 70 00
Germany, Feldkirchen bei Muenchen, (49) 089/90992-0
Israel, Tel-Aviv, (972) 03 498080
Italy, Assago, (39) (2) 575441
Netherlands, Rotterdam, (31) 10 407 11 11
Russia, Moscow, 007-095-4439785
Spain, Madrid, (34) (1) 308 2552
Sweden, Solna, (46) 8 705 5600
United Kingdom, Swindon, (44) (0793) 696000

EUROPEAN DISTRIBUTORS:

Austria, †*Bacher Electronics GmbH, Wien, (43) 1816020
Belgium, †*Inelco Distribution, Bruxelles, (32) 2 244 2811; *Diode Belgium, Zaventem, (32) 2 725 46 60
Denmark, *Avnet Nortec A/S, Herlev, (45) 4284 2000; †*ITT Multikomponent AS, Glostrup, (45) 4245 6645
Finland, †*OY Fintronic AB, Helsinki, (358) 0 682 791
France, *Arrow Electronique, Rungis Cedex, (33) (1) 4978 4978; †Metrologie, Asnieres Cedex, (33) (1) 4080 9000; *Tekelec, Sevres, (33) (1) 4623 2425
Germany, *Electronic 2000, Muenchen, (49) 89 42110-01; *Jermyn GmbH, Limburg, (49) 6431 5080; †Metrologie GmbH, Muenchen, (49) 89 724470; *Proelectron Vertriebs GmbH, Dreieich, (49) 6103 304343; †Rein Elektronik GmbH, Nettetal, (49) 2153 7330
Greece, †Ergodata, Kalithea, (30) 1 95 10 922; *Pouliadis Associates Corp., Athens, (30) 1 36 03 741
Ireland, †*Micro Marketing, Dublin, (353) (1) 298 9400
Israel, †*Eastronics Limited, Tel-Aviv, (972) 3 6458 777
Italy, *Intesi Div. Della Deutsche — Divisione ITT Industries GmbH, Assago (Milano), (39) 2 824701; *Lasi Elettronica, Milano, (39) 2 661431; †Omnilogic Telcom, Milano, (39) 2 48302640
Netherlands, †Datelcom, BD Maarsen, (31) 3465 95222; *Diode Components, NG Nieuwegein, (31) 3402 9 12 34; †*Koning en Hartman, AP Delft, (31) 15 609 906
Norway, *Avnet Nortec A/S, Hvalstad, (47) 284 6210; †Computer System Integration A/S, Skjetten, (47) 6 84 54 11
Portugal, *ATD Electronica LDA, Lisboa, (351) (1) 847 2200; †Metrologia Iberica Portugal, Lisboa, (351) (1) 847 2202

EUROPEAN DISTRIBUTORS (Contd.):

South Africa, †*EBE, Pretoria, (27) 12 803 7680-93
Spain, *ATD Electronica, Madrid, (34) (1) 661 6551; †Metrologia Iberica, Madrid, (34) (1) 661 1142
Sweden, †Avnet Computer AB, Farsta, (46) 8 705 18 00; *Avnet Nortec AB, Solna, (46) 8705 1800; *ITT Multikomponent AB, Solna, (46) 8 830020
Switzerland, †IMIC Microcomputer, Winkel-Ruti, (41) (1) 8620055; †*Industrade A.G., Wallisellen, (41) (1) 8328111
Turkey, *Empa Electronic, Istanbul, (90) (1) 599 3050
United Kingdom, *Arrow Electronics, Bedford, (44) 234 270272; *Avnet EMG Ltd., Hertsfordshire, (44) 462 488 500; *Bytech Components, Hants, (44) 256 707 107; †Bytech Systems, Berks, (44) 344 55 333; *Jermyn Electronics, Kent, (44) 732 743 743; †Metrologie VA, Bucks, (44) 494 526 271; *MMD/Rapid Ltd., Berks, (44) 734 750 697

INTERNATIONAL SALES OFFICES:

Australia, Intel Australia Pty. Ltd., Sydney, 61-2-975-3300; Intel Australia Pty. Ltd., Melbourne, 61-3-810-2141
Brazil, Intel Semiconductores do Brazil LTDA, Sao Paulo, 55-11-287-5899
China/Hong Kong, Intel PRC Corp., Beijing, (1) 500-4850; Intel Semiconductor Ltd., Hong Kong, (852) 844-4555
India, Intel Asia Electronics, Inc., Bangalore, 91-812-215065
Japan, Intel Japan K.K., Tsukuba HQ, 0298-47-8511; Intel Japan K.K., Tokyo HQ, 03-3201-3621; Intel Japan K.K., Tokyo, 03-3493-6081; Intel Japan K.K., Kanagawa, 045-474-7660; Intel Japan K.K., Osaka, 06-863-1091; Intel Japan K.K., Hachioji-shi, 0426-48-8770
Korea, Intel Korea, Ltd., Seoul, (2) 784-8186
Mexico, Intel Tecnologia de Mexico S.A. de C.V., Guadalajara, Jal., 523-640-1259
Singapore, Intel Singapore Technology, Ltd., (65) 250-7811
Taiwan, Intel Technology Far East Ltd., Taipei, 886-2-5144202

INTERNATIONAL DISTRIBUTORS:

Argentina, Dafsys S.R.L., Buenos Aires, 54.1334.1871
Australia, Email Electronics, Huntingdale, 011-61-3-544-8244; NSD-Australia, Victoria, 03 8909070
Brazil, Microlinear, Sao Paulo, 5511-220-2215
Chile, Sisteco, Santiago, 562-234-1644
China/Hong Kong, Novel Precision Machinery Co., Ltd., Kowloon, Hong Kong, (852) 360-8999
Guatemala, Abinitio, Guatemala City, 5022-32-4104
India, Priya International Limited, Bangalore, 91-812-214027, 812-214395; Priya International Limited, Bombay, 91-22-2863611, 22-2863676, 22-2863900, 22-2864026; Priya International Limited, New Delhi, 91-11-3314512, 11-3310413; Priya International Limited, Madras, 91-44-451031, 44-451597; Priya International Limited, Secunderabad, 91-842-813549, 842-813120; SES Computers and Technologies Pvt. Ltd., Delhi, 91-11-6849285, 11-6843006; SES Computers and Technologies Pvt. Ltd., Bombay, 91-22-4939977, 22-4943731; SES Computers and Technologies Pvt. Ltd., Bangalore, 91-812-348481, 812-343685
Jamaica, MC Systems, Kingston, (809) 926-0104
Japan, Asahi Electronics Co. Ltd., Kitakyushu-shi, 093-511-6471; CTC Components Systems Co., Ltd., Kanagawa 213, 044-852-5121; Dia Semicon Systems, Inc., Tokyo 154, 03-3439-1600; Okaya Koki, Nagoya-shi, 052-204-8315; Ryoyo Electro Corp., Tokyo 104, 03-3546-5011
Korea, J-Tek Corp., Seoul, (822) 557-8039; Samsung Electronics, Seoul, (822) 751-3680
Mexico, PSI S.A. de C.V., Cuernavaca-MOR, 52-73-11-1994/5
New Zealand, Email Electronics, Penrose, Auckland, 011-64-9-591-155
Saudi Arabia, AAE Systems, Inc., Sunnyvale, CA U.S.A., (408) 732-1710
Singapore, Electronic Resources Pte, Ltd., (65) 283-0888
South Africa, Electronic Building Elements, Pretoria, 011-2712-803-7680
Taiwan, Micro Electronics Corp., Taipei, R.O.C., (886) 2-7198419; Acer Sertek, Inc., Taipei, R.O.C., (886) 2-501-0055
Uruguay, Interfase, Montevideo, 5982-49-4600
Venezuela, Unixel C.A., Caracas, 582-238-7749

*Technical Distributor
†VAD